DISCUSSION FOCUS

Why is data modeling so important to the database designer?

We are said to live in an information age, and data constitute the most basic information units employed by an information system. Data modeling provides a way to reconcile the very different end-user views of the nature and role of data.

A data model is an abstraction, providing a relatively simple representation of complex real-world data structures. The model's function is thus to help us understand the complexity of a real-world data environment. Such understanding is likely to yield useful solutions to the problems inherent in creating, using, and managing data.

If a database is to be useful and flexible, it must be well designed. The database design process must be based on an appropriate data model if it is to yield a proper blueprint of the design.

What are business rules, what is their source, and why are they crucial?

Business rules are precise statements, derived from a detailed description of the organization's operations. *When written properly*, business rules define one or more of the following modeling components:

- entities
- relationships
- attributes
- connectivities
- cardinalities
- constraints

Because the business rules form the basis of the data modeling process, their precise statement is crucial to the success of the database design. And, because the business rules are derived from a precise description of operations, much of the design's success depends on the accuracy of the description of operations.
Examples of business rules are:

- An invoice contains one or more invoice lines.
- Each invoice line is associated with a single invoice.
- A store employs many employees.
- Each employee is employed by only one store.
- A college has many departments.
- Each department belongs to a single college. (This business rule reflects a university that has multiple colleges such as Business, Liberal Arts, Education, Engineering, etc.)
- A driver may be assigned to drive many different vehicles.
- Each vehicle can be driven by many drivers. (Note: Keep in mind that this business rule reflects the assignment of drivers during some period of time.)
- A client may sign many contracts.
- Each contract is signed by only one client.
- A sales representative may write many contracts.
- Each contract is written by one sales representative.

Note that each relationship definition requires the definition of two business rules. For example, the relationship between the INVOICE and (invoice) LINE entities is defined by the first two business rules in the bulleted list. This two-way requirement exists because there is always a two-way relationship between any two related entities. (This two-way relationship description also reflects the implementation by many of the available CASE tools.)

Keep in mind that the ER diagrams cannot always reflect all of the business rules. For example, examine the following business rule:

A customer cannot be given a credit line over $10,000 unless that customer has maintained a satisfactory credit history (as determined by the credit manager) during the past two years.

This business rule describes a constraint that cannot be shown in the ER diagram. The business rule reflected in this constraint would be handled at the applications software level through the use of a trigger or a stored procedure.
How are M:N relationships handled in the development of an ER diagram?

Although M:N relationships may properly be viewed in a relational database model at the conceptual level, such relationships should not be implemented, because their existence creates undesirable redundancies. Therefore, M:N relationships must be decomposed into 1:M relationships to fit into the ER framework. For example, if you were to develop an ER model for a video rental store, you would note that tapes can be rented more than once and that customers can rent more than one tape.

To make the discussion more reasonable keep in mind that newly arrived tapes that have just been entered into inventory have not yet been rented and that some tapes may never be rented at all if there is no demand for them. Therefore, CUSTOMER is optional to TAPE. Assuming that the video store only rents videos and that a CUSTOMER entry will not exist unless a person coming into the video store actually rents that first tape, TAPE is mandatory to CUSTOMER. Note that this discussion includes a very brief description of the video store's operations and some business rules. The relationship between customers and tapes would thus be M:N, as shown in Figure IM4.1.

**Figure IM4.1 The M:N Relationship**

- A customer can rent many tapes.
- A tape can be rented by many customers.
- Some customers do not rent tapes.
  
  (Such customers might buy tapes or other items.) Therefore, TAPE is optional to CUSTOMER in the rental relationship.
- Some tapes are never rented.
  
  Therefore, CUSTOMER is optional to TAPE in the rental relationship.

![Chen Model](image)

![Crow's Foot Model](image)
The M:N relationship depicted in Figure IM4.1 must be broken up into two 1:M relationships through the use of a bridge entity, also known as a composite entity. The composite entity, named RENTAL in the example shown in Figure IM4.2, is composed of at least the primary key components of the two entities it bridges, because the RENTAL entity’s foreign keys must point to the primary keys of the two entities CUSTOMER and TAPE. And, because a tape that is never rented will never show up in the RENTAL entity, RENTAL has become optional to TAPE.

That's why the optional symbol has migrated from CUSTOMER to the opposite side of RENTAL. Also note that the relationship name is often written above the relationship line, rather than breaking the relationship line as was shown in Figure IM4.1.

Figure IM4.2 Decomposition of the M:N Relationship

Chen Model

Crow’s Foot Model

Because the M:N relationship has now been decomposed, the ER model shown in Figure IM4.2 is implementable.

Although the Chen ERD in Figure IM4.2 is acceptable from a conceptual point of view, some data modeling software requires that the model be further refined to include the names of the relationships between the composite entity and the entities for which it serves as the bridge. Therefore, the previous ER diagram may be further refined as shown in Figure IM4.3. Note that the Chen Model now is the equivalent of the Crow’s Foot Model in Figure IM4.2.
Remind the students that the relationships are read from the 1 side to the M side. Therefore, the relationships are read as:

CUSTOMER generates RENTAL
TAPE enters RENTAL

Although the ERDs shown in Figures IM4.2 and IM4.3 depict the relationships well at the purely conceptual level, the nature of the relationships and their implementation become more obvious if you define the attributes that describe each entity.

The ERD is shown in Figure IM4.4.

In Figure IM4.4 note the following features:

- The dashed relationship lines indicate weak relationships. In this case, the RENTAL entity’s primary key is RENT_NUM – and this PK did not use any attribute from the CUSTOMER and TAPE entities.
- The RENTAL entity’s PK could have been the combination TAPE_CODE + CUS_NUM. This composite PK would have created strong relationships between RENTAL and CUSTOMER and between RENTAL and TAPE. Because this composite PK was not used, it is a candidate key.
- In this case, the designer made the decision to use a single-attribute PK rather than a composite PK. However, it is useful to point out that single-attribute PKs are usually more desirable than composite PKs – especially when relationships must be established between the RENTAL and some – as yet unidentified – entity.
Entity Relationship (ER) Modeling

cannot use a composite PK as a foreign key in a related entity!

The cardinalities reflect the rental transactions. Each rental transaction, i.e., each record in the RENTAL table, will reference one and only one customer and one an only one tape. The (simplified!) implementation of this model may thus yield the sample data shown in the database in Figure IM4.5. The database's relational schema is shown in Figure IM4.6.

**Figure IM4.5 The Ch04_Rental Database Tables**

![Database Tables](image)

The relational schema that corresponds to the design in

**Figure IM4.6 The Ch04_Rental Database Relational Schema**

![Relational Schema](image)
The Ch04_Rental database’s TAPE and RENTAL tables contain some attributes that merit additional discussion.

- The TAPE_CODE attribute values include a “trailer” after the dash. For example, note that the third record in the TAPE table has a PK value of R2345-2. The “trailer” indicates the tape copy. So why include a separate TAPE_COPY attribute? This decision was made to make it easier to generate queries that make use of the tape copy value. (It’s much more difficult to use a right-string function to “strip” the tape copy value than simply using the TAPE_COPY value. And “simple” usually translates into “fast” in a query environment – “fast” is a good thing!

- The RENTAL table uses two dates: RENT_DATE_OUT and RENT_DATE_RETURN. This decision leaves the RENT_DATE_RETURN value null until the tape is returned. Naturally, such nulls can be avoided by creating an additional table in which the return date is not a named attribute. Note the following few check-in and check-out transactions:

<table>
<thead>
<tr>
<th>RENT_NUM</th>
<th>TRANS_TYPE</th>
<th>TRANS_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10050</td>
<td>Checked-out</td>
<td>10-Jan-2004</td>
</tr>
<tr>
<td>10050</td>
<td>Returned</td>
<td>11-Jan-2004</td>
</tr>
<tr>
<td>10051</td>
<td>Checked-out</td>
<td>10-Jan-2004</td>
</tr>
<tr>
<td>10051</td>
<td>Returned</td>
<td>11-Jan-2004</td>
</tr>
<tr>
<td>10052</td>
<td>Checked-out</td>
<td>11-Jan-2004</td>
</tr>
<tr>
<td>10053</td>
<td>Returned</td>
<td>10-Jan-2004</td>
</tr>
</tbody>
</table>

The decision to leave the RENT_DATE_RETURN date in the RENTAL table – and leaving its value null until the tape is returned is – again – up to the designer, who evaluates the design according to often competing goals: simplicity, elegance, reporting capability, query speed, index management, and so on.

**Discuss why Figure IM4.6's database is not quite ready for prime time.** For example, its structure allows customers to rent only one tape per rental transaction. Therefore, you'd have to generate a separate rental transaction for each tape rented by a customer. (In other words, if a customer rents five tapes at a time, you'd have to generate five separate rentals.) Clearly, the design would be much improved by expanding it to include rental lines, as is done in Figure IM4.7.
Figure IM4.7 An Implementable Video Rental ERD

What role does the ER diagram play in the design process?

A completed ER diagram is the actual blueprint of the database. Its composition must reflect an organization's operations accurately if the database is to meet that organization's data requirements. It forms the basis for a final check on whether the included entities are appropriate and sufficient, on the attributes found within those entities, and on the relationships between those entities. It is also used as a final crosscheck against the proposed data dictionary entries. The completed ER diagram also lets the designer communicate more precisely with those who commissioned the database design. Finally, the completed ER diagram serves as the implementation guide to those who create the actual database. In short, the ER diagram is as important to the database designer as a blueprint is to the architect and builder.
Why is the discovery of entity types and subtypes important to the designer?

The presence of nulls in a database table is undesirable and should be tolerated only if it cannot be avoided.

1. Suppose you are working within the framework of the conceptual model in Figure Q4.5.

![Figure Q4.5 The Conceptual Model for Question 5](image)

Given the conceptual model in Figure Q4.5,

a. Write the business rules that are reflected in it.

Even a simple ERD such as the one shown in Figure Q4.5 is based on many business rules. Make sure that each business rule is written on a separate line and that all of its details are spelled out. In this case, the business rules are derived from the ERD in a "reverse-engineering" procedure designed to document the database design. In a real world database design situation, the ERD is generated on the basis of business rules that are written before the first entity box is drawn. (Remember that the business rules are derived from a carefully and precisely written description of operations.)

Given the ERD shown in Figure Q4.5, you can identify the following business rules:
1. A customer can own many cars.
2. Some customers do not own cars.
3. A car is owned by one and only one customer.
4. A car can get (generate) a maintenance record more than once.
5. Each maintenance record is generated by one and only one car.
6. Some cars have not (yet) generated a maintenance procedure.
7. Each maintenance procedure can use many parts.
    (Comment: A maintenance procedure may include multiple maintenance
    actions, each one of which may or may not use parts. For example, 10,000-mile
    check may include the installation of a new oil filter and a new air filter. But
    tightening an alternator belt does not require a part.)
8. A part may be used in many maintenance records.
    (Comment: Each time an oil change is made, an oil filter is used. Therefore,
    many oil filters may be used during some period of time. Naturally, you are not
    using the same oil filter each time – but the part classified as “oil filter” shows
    up in many maintenance records as time passes.)
b. Identify all the cardinalities.

Given the ERD shown in Figure Q4.5, the cardinalities are shown in Figure Q4.5b1.

**Figure Q4.5b1 The Chen ERD with Cardinalities Shown**

![Chen ERD with Cardinalities Shown](image)

The Crow’s Foot ERD, shown in Figure Q4.5b2, does not show cardinalities directly. Instead, the cardinalities are implied through the Crow’s Foot symbols. Note that Visio does not generate M:N relationship, because such relationships are inappropriate. (M:N relationships should never be implemented because they generate undesirable data redundancies that trigger data anomalies. To show the M:N relationship anyway, we have simply generated two relationship lines between MAINTENANCE and PART. Only one of the two relationships can be anchored to the entity.

**Figure Q4.5b2 The Crow’s Foot ERD with Cardinalities Implied**

![Crow’s Foot ERD with Cardinalities Implied](image)
c. Explain how you would modify the design to remove the M:N relationship between MAINTENANCE and PART.

The M:N relationship is removed through the creation of two 1:M relationships. In this case, we have created a new entity named LINE to reflect the fact that each maintenance record may include more than one procedure. For example, a maintenance record may include an oil change, tire balancing, and the replacement of a clogged fuel filter. Note that this design feature reflects the same relationships that your students encountered when they created an invoice with multiple invoice lines – with each invoice line referencing a product. (See Figure 4.33 in the text.) The results of the modification are shown in Figure Q4.5c1.

Figure Q4.5c1 The Modified Crow’s Foot ERD

As you discuss the ERD shown in Figure Q4.5c1 with your students, note the following:

- The optionality symbol next to the LINE entity indicates that some parts may never be used in maintenance. (Sooner or later, such unused parts are scrapped.)
- Actually, PART should be shown as optional to LINE because some maintenance procedures, such as tightening a bolt, do not require the use of parts. However, the optionality on the PART side cannot be shown until the PART_CODE attribute is included in the LINE entity and this PART-CODE is marked as “not required” in the attribute definition.
- The “contains” relationship between MAINTENANCE and LINE is shown as strong. (The relationship line is solid.) This reflects the likely primary key definition of the LINE entity. In this case, the PK is probably defined by MAINT_NUM + LINE_NUM.
You can also show the effect of the modification in the Chen entity. Note the cardinality definitions in Figure Q4.5c2.

**Figure Q4.5c2 The Modified Chen ERD**

Remind your students that the cardinalities shown in a Chen ERD always point to the number of occurrences in the related table. For example, the cardinality (1,N) written next to the MAINTENANCE entity indicates that the maintenance record reference will occur one or more times in the LINE table. For example, take a look at the (simplified) contents of the following MAINTENANCE and LINE tables and note that the MAINT_NUM 10001 occurs three times in the LINE table:

<table>
<thead>
<tr>
<th>MAINTENANCE</th>
<th>LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINT_NUM</td>
<td>LINE_NUM</td>
</tr>
<tr>
<td>10001</td>
<td>1</td>
</tr>
<tr>
<td>10001</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAINTENANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINT_NUM</td>
</tr>
<tr>
<td>10001</td>
</tr>
<tr>
<td>10002</td>
</tr>
<tr>
<td>10003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAINTENANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINT_DATE</td>
</tr>
<tr>
<td>15-Mar-2004</td>
</tr>
<tr>
<td>15-Mar-2004</td>
</tr>
<tr>
<td>16-Mar-2004</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>10001</td>
</tr>
<tr>
<td>10002</td>
</tr>
<tr>
<td>10003</td>
</tr>
<tr>
<td>10003</td>
</tr>
</tbody>
</table>

2. **When might you want to use entity supertypes and subtypes and what must be the relationship between them? Give an example.**

Entity supertypes and subtypes are generally used to avoid the proliferation of nulls. Review the text’s section 4.1.10 (Entity supertypes and subtypes.) and note the illustration of the supertype/subtype use with reference to Figures 4.27, 4.28, and 4.29. The relationship between a supertype and its subtypes is 1:1.

3. **How would you (graphically) identify each of the following ER model components?**

   **a. an entity**

   In both the Chen Model and the Crow’s Foot model, an entity is represented by a rectangle containing the entity name. (Remember that, in ER modeling, the word "entity" actually refers to the entity set.)

   In the Chen model, a **weak entity** is indicated through the use of a double-lined rectangle and a **composite entity** is identified through a diamond within a rectangle. (See Figure Q4.7a.)

   **Figure Q4.7a Entity Representation in a Chen ERD**

   ![Figure Q4.7a Entity Representation in a Chen ERD](image)

   The Crow’s Foot ERD – as represented in Visio Professional – does not distinguish among the entities. Instead, the Crow’s Foot ERD uses relationship types – strong or weak – to indicate the nature of the relationships between entities. For example, a strong relationship may be used to imply the existence of a weak entity.

   A composite entity is defined by the fact that at least one of the PK attributes is also a foreign key. Therefore, the Visio Crow’s Foot ERD’s composite and weak entities
are not differentiated – whether or not an entity is weak or composite depends on the definition of the business rule(s) that describe the relationships. In any case, two conditions must be met before an entity can be classified as weak:
1. The entity must be existence-dependent on its parent entity
2. The entity must inherit at least part of its primary key from its parent entity.

b. the cardinality (0,N)

There are several examples of this cardinality in Figure Q4.5c2. Note particularly the (0,N) cardinalities written next to CUSTOMER, CAR, and PART.

c. a weak relationship

A weak relationship is indicated by a dashed line in the (Visio) ERD. The Chen ERD does not make a distinction between relationship types. However, the absence of a weak entity or a composite entity implies the existence of a weak relationship in the Chen ERD.

d. a strong relationship

A strong relationship is indicated by a solid line in the (Visio) ERD. The Chen ERD does not make a distinction between relationship types. However, the presence of a weak entity or a composite entity implies the existence of a strong relationship in the Chen ERD.

4. The Hudson Engineering Group (HEG) has contacted you to create a conceptual model whose application will meet the expected database requirements for its training program. The HEG administrator gives you the following description of the training group’s operating environment:

The HEG has 12 instructors and can handle up to 30 trainees per class. HEG offers five “advanced technology” courses, each of which may generate several classes. If a class has fewer than 10 trainees in it, it will be canceled. It is, therefore, possible for a course not to generate any classes. Each class is taught by one instructor. Each instructor may teach up to two classes or may be assigned to do research only. Each trainee may take up to two classes per year.
Given this information, do the following:

a. Define all the entities and relationships. (Use Table 4.2 as your guide.)
The HEG entities and relationships are shown in Table Q4.8a.

Table Q4.8a The Components of the HEG ERD

<table>
<thead>
<tr>
<th>ENTITY</th>
<th>RELATIONSHIP</th>
<th>CONNECTIVITY</th>
<th>ENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTRUCTOR</td>
<td>teaches</td>
<td>1:M</td>
<td>CLASS</td>
</tr>
<tr>
<td>COURSE</td>
<td>generates</td>
<td>1:M</td>
<td>CLASS</td>
</tr>
<tr>
<td>CLASS</td>
<td>is listed in</td>
<td>1:M</td>
<td>ENROLL</td>
</tr>
<tr>
<td>TRAINEE</td>
<td>is written in</td>
<td>1:M</td>
<td>ENROLL</td>
</tr>
</tbody>
</table>

As you examine the summary in Table Q4.8a, it is reasonable to assume that many of the relationships are optional and that some are mandatory. (Remember a point we made earlier: when in doubt, assume an optional relationship.)

- A COURSE does not necessarily generate a class during each training period. (Some courses may be taught every other period or during some other specified time frames. Therefore, it is reasonable to assume that CLASS is optional to COURSE.

- Each CLASS must be related to a COURSE. (The class must cover designated course material!) Therefore, COURSE is mandatory to CLASS.

- Some instructors may teach a class every other period or even rarely. Therefore, it is reasonable to assume that CLASS is optional to INSTRUCTOR during any enrollment period. This optionality makes sense from an implementation point of view, too. For example, if you appoint a new instructor, that instructor will not – yet – have taught a class.

- Not all trainees are likely to be enrolled in classes during some time period. In fact, in a real world setting, many trainees are likely to get informal “on the job” training without going to formal classes. Therefore, it is reasonable to assume that ENROLL is optional to TRAINEE.

- You cannot create an enrollment record without having a trainee. Therefore, TRAINEE is mandatory to ENROLL. (Discussion point: What about making TRAINEE optional to ENROLL? In any case, optional relationships may be used for operational reasons, whether or not they are directly derived from a business rule.)

Note that a real world database design requires the explicit recognition of each relationship’s characteristics. When in doubt, ask the end users!
b. Describe the relationship between instructor and class in terms of connectivity, cardinality, and existence-dependence.

Both questions a and b have been addressed in the ER diagram shown in Figure Q4.8b1. As you discuss Figure Q4.8b1, keep the discussion in part (a) in mind. Also, note the following points:

- A trainee can take more than one class, and each class contains many (10 or more) trainees, so there is a M:N relationship between TRAINEE and CLASS. (Therefore, a composite entity is used to serve as the bridge between TRAINEE and CLASS.)
- A class is taught by only one instructor, but an instructor can teach up to two classes. Therefore, there is a 1:M relationship between INSTRUCTOR and CLASS.
- Finally, a COURSE may generate more than one CLASS, while each CLASS is based on one COURSE, so there is a 1:M relationship between COURSE and CLASS.

These relationships are all reflected in the ER diagram shown in Figure Q4.8b1. Note the optional and mandatory relationships:

- To exist, a CLASS must have TRAINEEs enrolled in it, but TRAINEEs do not necessarily take CLASSes. (Some may take "on the job training.")
- An INSTRUCTOR may not be teaching any CLASSes during some enrollment periods. For example, an instructor may be assigned to duties other than training. However, each CLASS must have an INSTRUCTOR.
- If not enough people sign up for a CLASS, a COURSE may not generate any CLASSes, but each CLASS must represent a COURSE.
Figure Q4.8b1 The Chen ERD for HEG

- **INSTRUCTOR** teaches **CLASS**
  - M: (1,1)
  - M: (10,30)
  - (0,2)

- **CLASS** generates **COURSE**
  - M: (1,1)
  - (0,N)

- **TRAINEE**
  - M: (1,1)
  - (0,2)

- **ENROLL**